DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

perceptrons

Fernando Berzal, berzal@acm.org

perceptrons

Introduction

Artificial neural networks models

Model of artificial neuron networks

Perceptron activation functions

Neuron McCulloch and Pitts The perceptron learning

algorithm Geometric interpretation Limitations

The multiclass Perceptron

Introduction

Artificial Neural Networks [RNA]

Artificial Neural Networks [ANN]

Network elements or simple processing units (EP / UP in Spanish or PE / PU in English), interconnected by synaptic connections,

wherein each connection has a weight (force) that is set from experience (data).

Introduction

Artificial Neural Networks [RNA]

Artificial Neural Networks [ANN]

Artificial Neuron model:

Simple calculation model. Each neuron receives stimuli from other neurons, adds and transmits a response according to its activation function.

Neural Network Model [topology]:

Structure of the neural network.

Organization and number of neurons and connections.

Introduction

Artificial Neural Networks [RNA]

Artificial Neural Networks [ANN]

Artificial neural networks provide a model of parallel computing and distributed able to learn from examples (data)

The **learning algorithms (** associated with particular network models)
Allow go changing the weights of the synaptic connections so that the network learn from the examples presented.

Introduction

Artificial Neural Networks [RNA]

Artificial Neural Networks [ANN]

They are not programmed, train.

Examples need to have, in a sufficient number and a representative distribution to be able to generalize properly.

They require a validation process to assess the "quality" of learning achieved.

Introduction

Artificial Neural Networks [RNA]

Artificial Neural Networks [ANN]

We'll see how ...

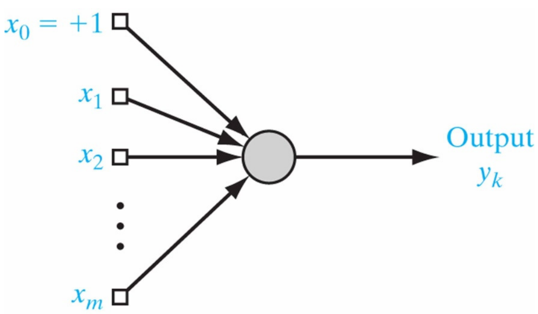
Training artificial neural networks (for different network models).

Preparing (preprocessing) Examples necessary for training.

Assess the quality of the learning process.

Introduction: Network Models

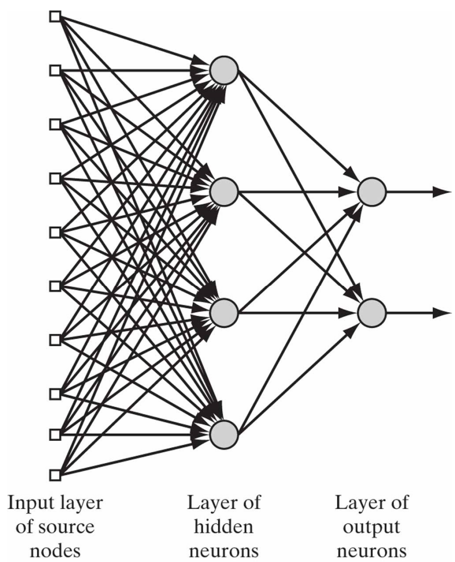
perceptron



[Haykin, "Neural Networks and Learning Machines", 3rd edition]

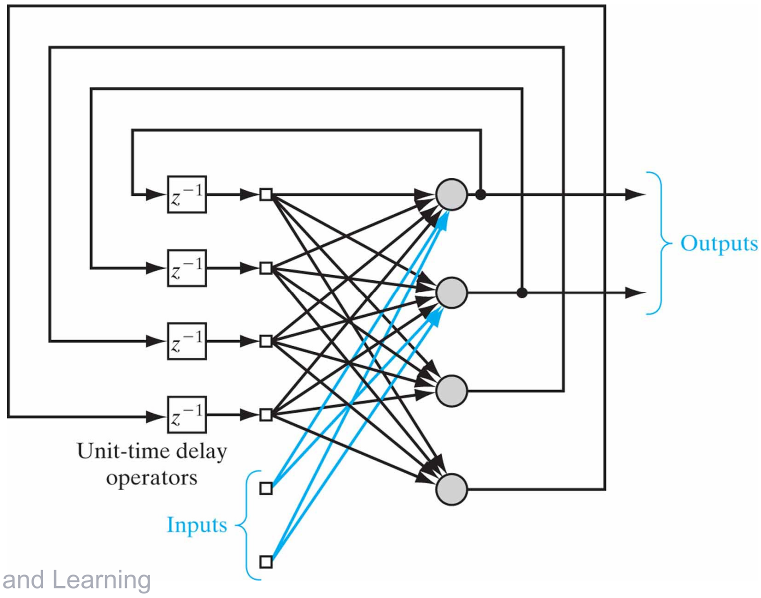
Feed-forward networks

(Topology layers)



Introduction: Network Models

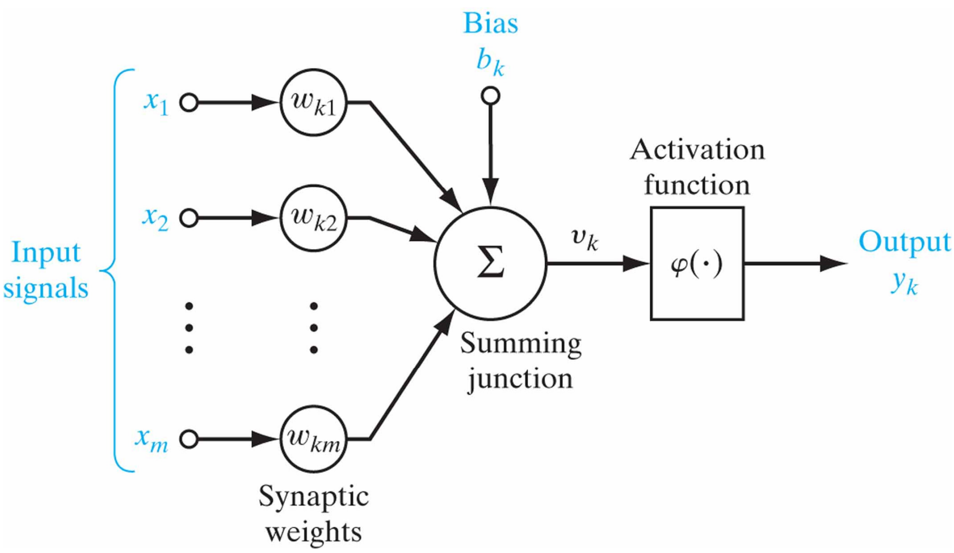
recurrent networks



[Haykin, "Neural Networks and Learning Machines", 3 rd edition]

Artificial neural model

nonlinear model of an artificial neuron

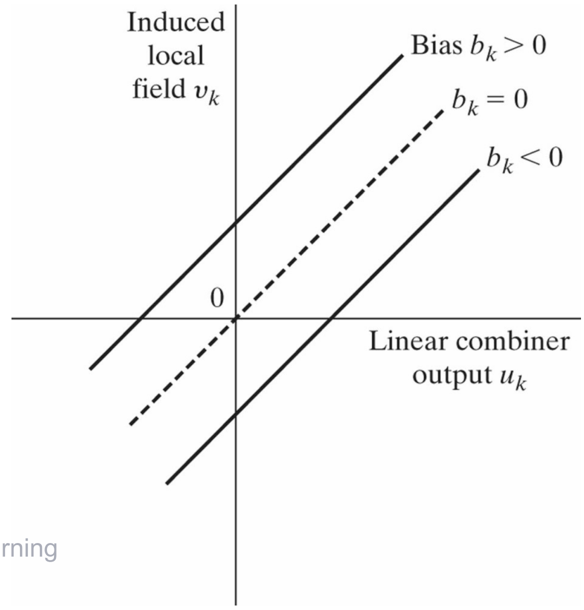


[Haykin, "Neural Networks and Learning Machines", 3 rd edition]

Artificial neural model

affine transformation caused by the presence of **bias b_k [Bias]**

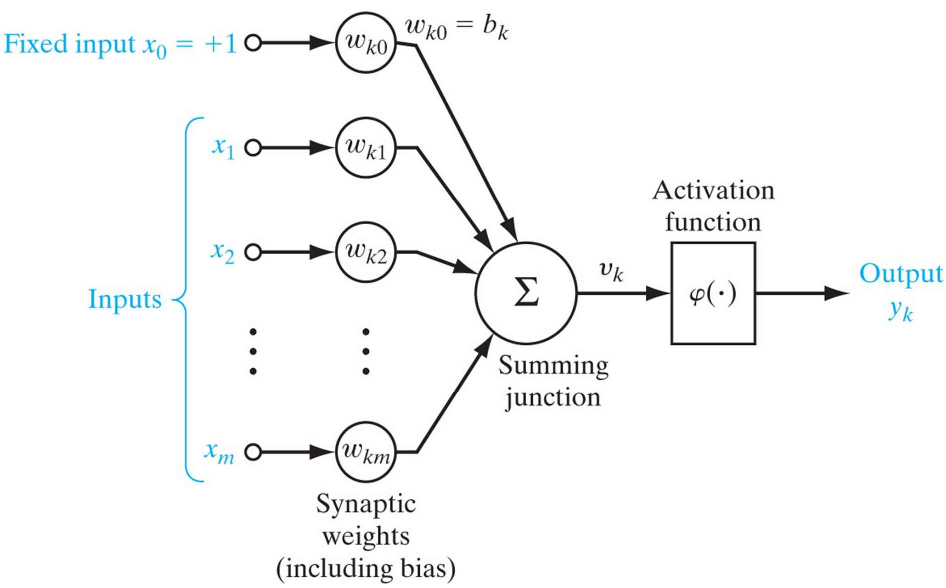
$v_k = b_k$ when $U_k = 0$



[Haykin, "Neural Networks and Learning Machines", 3rd edition]

Artificial neural model

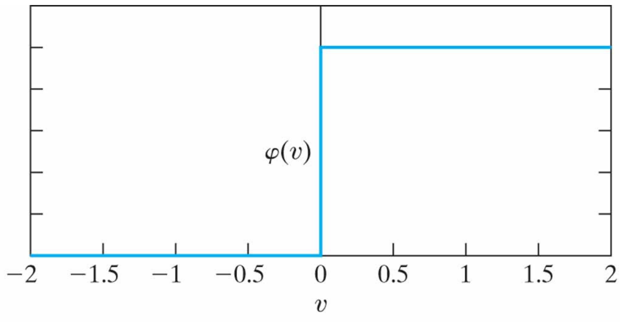
nonlinear model of an artificial neuron ($w_{k0} = b_k$)



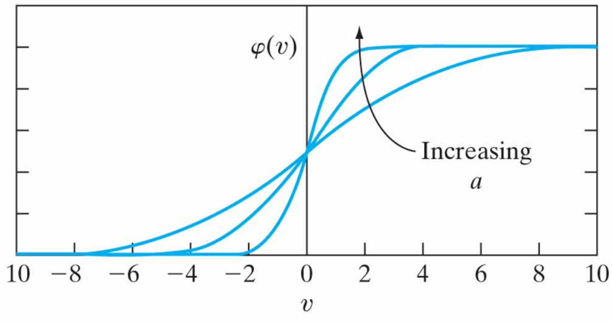
[Haykin, "Neural Networks and Learning Machines", 3rd edition]

Artificial neural model

Activation functions φ φφ φ (v)



binary neurons



sigmoidal neurons

[Haykin, "Neural Networks and Learning Machines", 3rd edition]

Artificial neuron model

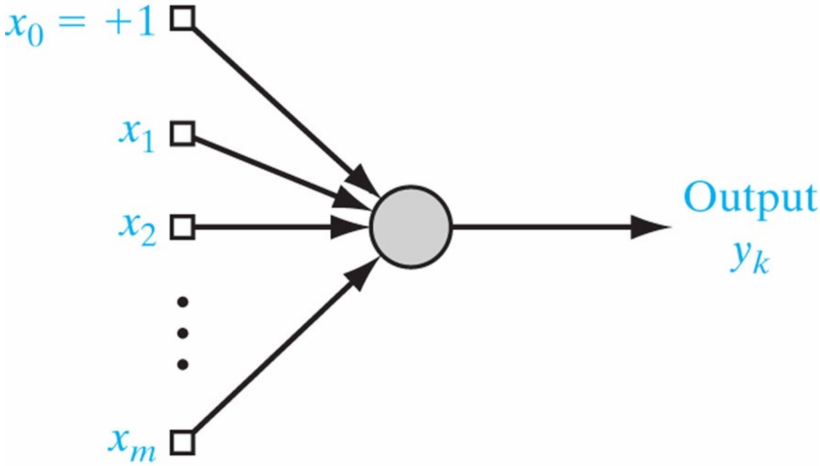
Sigmoidal activation functions

logistic function [0,1]

$$\varphi(v) = \frac{\text{one}}{\text{one} + e^{-av}}$$

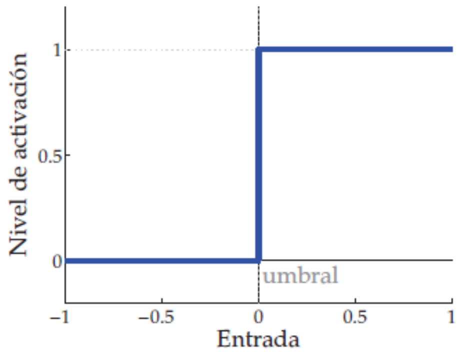
Hyperbolic Tangent [-1,1]

$$\varphi(v) = \tanh v = \frac{e^{v} - e^{-v}}{e^{v} + e^{-v}}$$



Neuron model

binary threshold neurons [McCulloch & Pitts, 1943]



$$= \sum {}_i W X_i Z$$

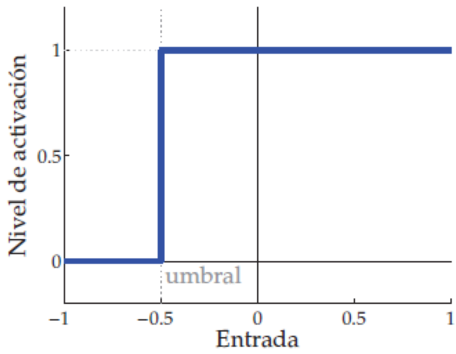
$$= \begin{cases} z \text{ and } & 1 \quad \text{if} \geq 0 \\ 0 & \text{anothe in case} \end{cases}$$

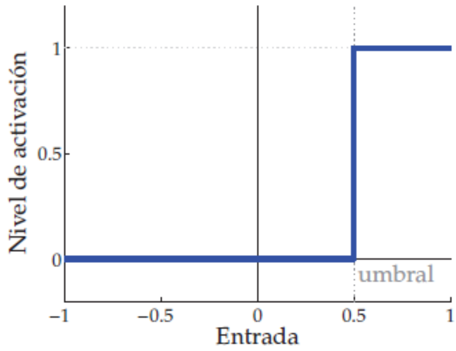
assuming $x_0 = 1$ and $w_0 = b$ (threshold $\theta = - b$)

perceptrons

Neuron model

binary threshold neurons



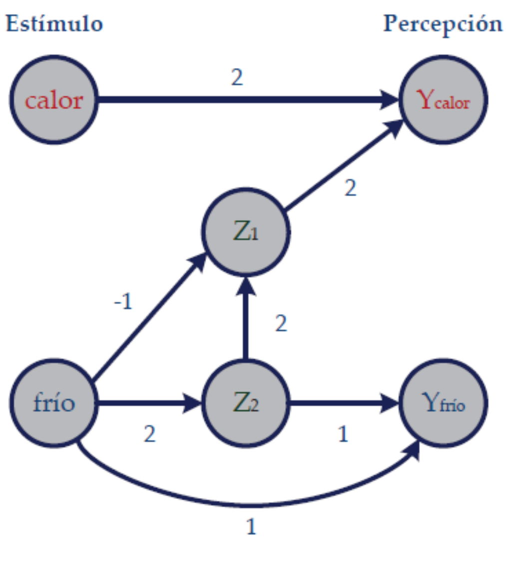


Threshold effect

perceptrons

Neuron model

Example: physiological perception of heat and cold



perceptrons

The first generation of neural networks

Popularized by Frank Rosenblatt in the 60s Minsky and Papert analyzed what they could do and showed its limitations in his 1969 book.

Many thought that these limitations were extended to all models of neural networks, although it is not.

Its learning algorithm is still used for tasks in which the feature vectors contain millions of items.

perceptrons

Pattern Recognition

input data are converted into a feature vector x_{j}.

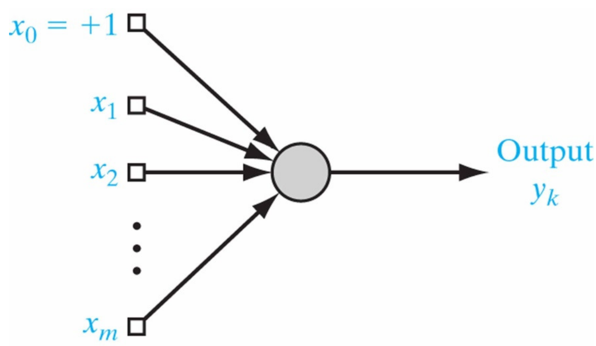
associated weights are learned each of these characteristics for a scalar from each input vector value.

If this scalar value is above a threshold, it is decided that the input vector corresponds to an example of the target class (and $_{k}$ = one).

perceptrons

Learning algorithm

A (positive) threshold is equivalent to a bias / bias (negative), so we can avoid dealing separately adding the threshold a fixed input $x_0 = +1$. Thus, the threshold learn as if a weight more.



perceptrons

Learning algorithm

Examples of the training set are selected using any policy to ensure that all training examples will end up choosing:

If the output is correct, the weights as they left. If the output unit incorrectly gives a zero, the input vector to the vector of weights is added. If the output unit incorrectly gives a one, the input vector weight vector is subtracted.

perceptrons

TABLE 1.1 Summary of the Perceptron Convergence Algorithm

Variables and Parameters:

$\mathbf{x}(n) = (m+1)$-by-1 input vector
$= [+1, x_1(n), x_2(n), \ldots, x_m(n)]^T$
$\mathbf{w}(n) = (m+1)$-by-1 weight vector
$= [b, w_1(n), w_2(n), \ldots, w_m(n)]^T$
b = bias
$y(n)$ = actual response (quantized)
$d(n)$ = desired response
η = learning-rate parameter, a positive constant less than unity

1. *Initialization.* Set $\mathbf{w}(0) = \mathbf{0}$. Then perform the following computations for time-step $n = 1, 2, \ldots$.
2. *Activation.* At time-step n, activate the perceptron by applying continuous-valued input vector $\mathbf{x}(n)$ and desired response $d(n)$.
3. *Computation of Actual Response.* Compute the actual response of the perceptron as
$$y(n) = \operatorname{sgn}[\mathbf{w}^T(n)\mathbf{x}(n)]$$
where $\operatorname{sgn}(\cdot)$ is the signum function.
4. *Adaptation of Weight Vector.* Update the weight vector of the perceptron to obtain
$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta[d(n) - y(n)]\mathbf{x}(n)$$
where
$$d(n) = \begin{cases} +1 & \text{if } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_1 \\ -1 & \text{if } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_2 \end{cases}$$
5. *Continuation.* Increment time step n by one and go back to step 2.

[Haykin, "Neural Networks and Learning Machines", 3 rd edition]

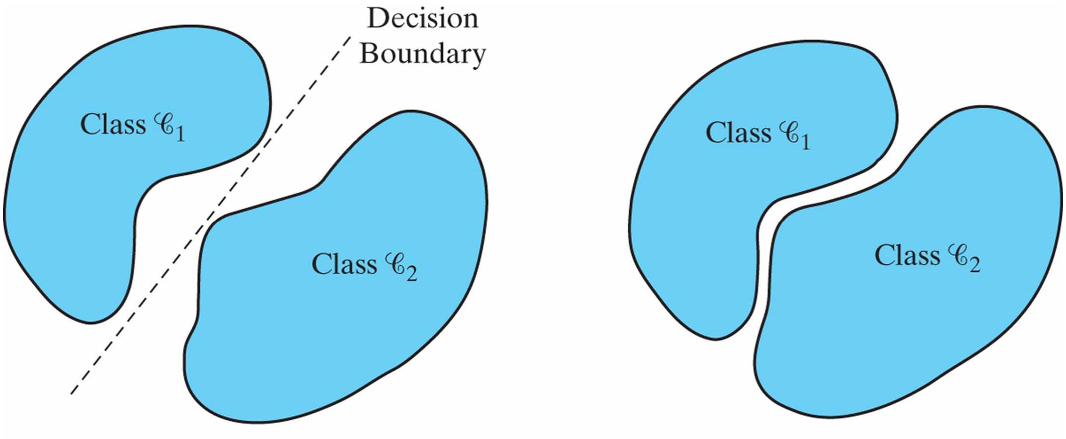
perceptrons

Learning algorithm

The perceptron learning algorithm is guaranteed to find a set of weights to provide the correct answer if such a set exists .

The perceptron is a model of linear classification, which will be able to correctly classify input examples provided the classes are linearly separable.

perceptrons



Lessons

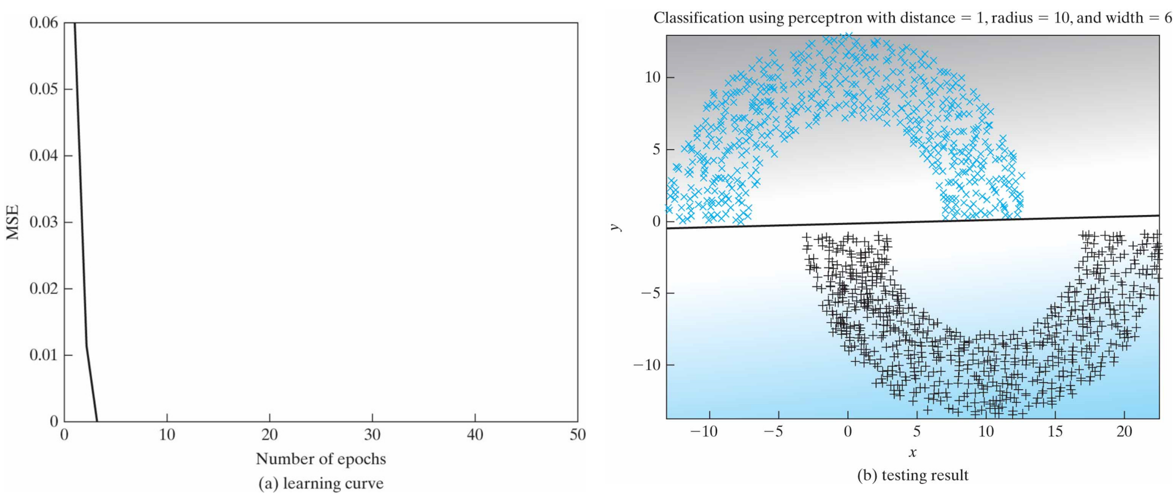
linearly separable

Lessons

linearly inseparable

[Haykin, "Neural Networks and Learning Machines", 3 rd edition]

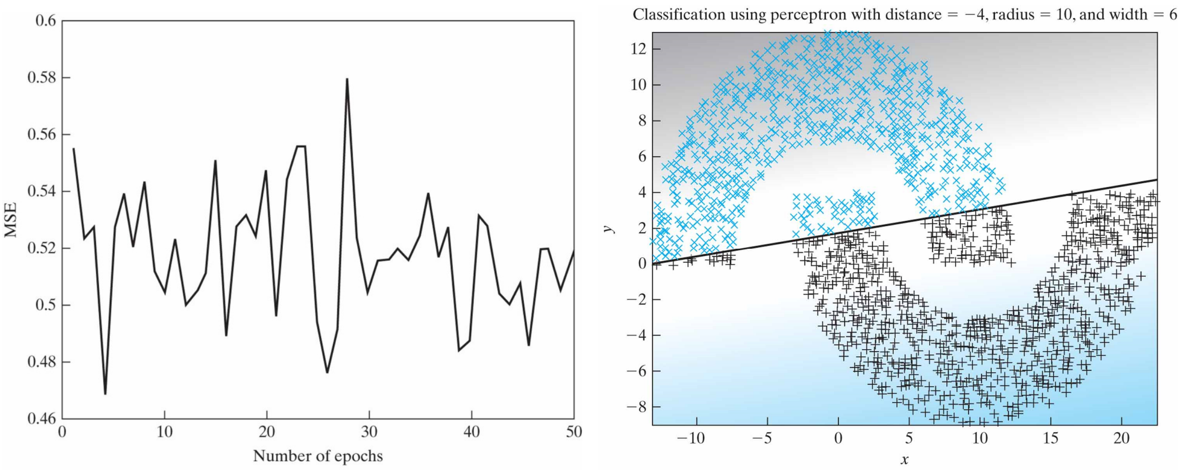
perceptrons



linearly separable classes

[Haykin, "Neural Networks and Learning Machines", 3 rd edition]

perceptrons



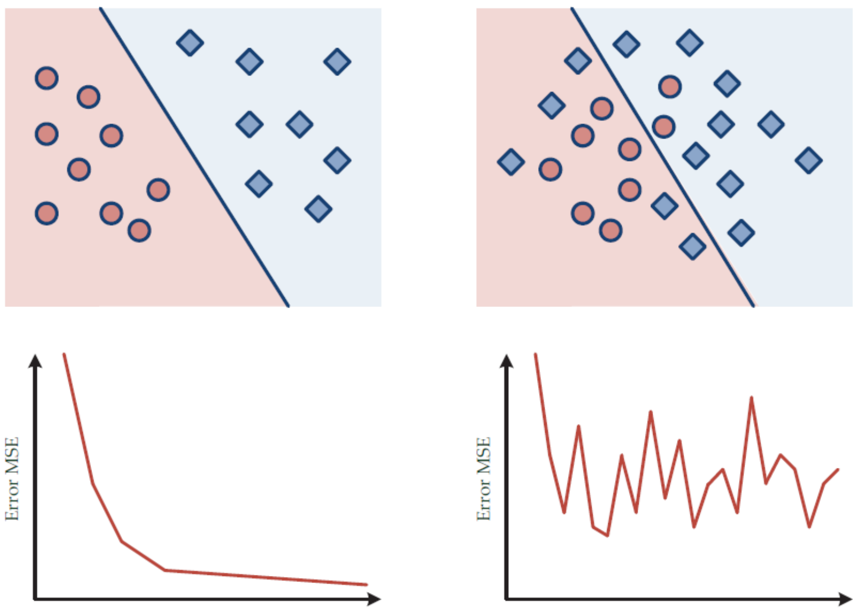
(a) learning curve (b) testing result

linearly non-separable classes

[Haykin, "Neural Networks and Learning Machines", 3 rd edition]

perceptrons

linearly separable classes vs. not linearly separable classes



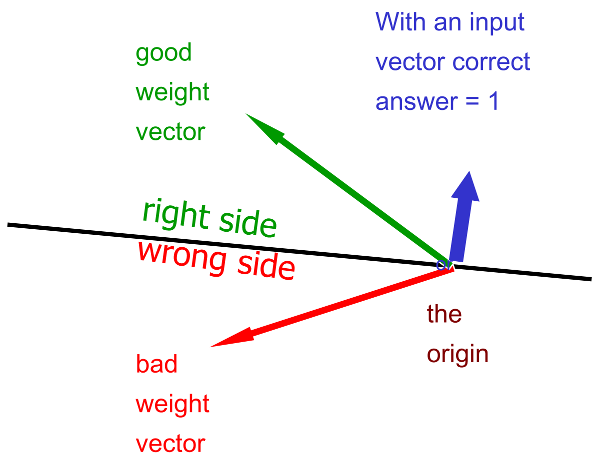
Learning algorithm: Geometric interpretation

Weight space:

A dimension for each weight. Each point in space represents a particular value for the set of weights. Training each case corresponds to a hyperplane passing through the origin

(After removing the threshold and include it as another burden)

Learning algorithm: Geometric interpretation

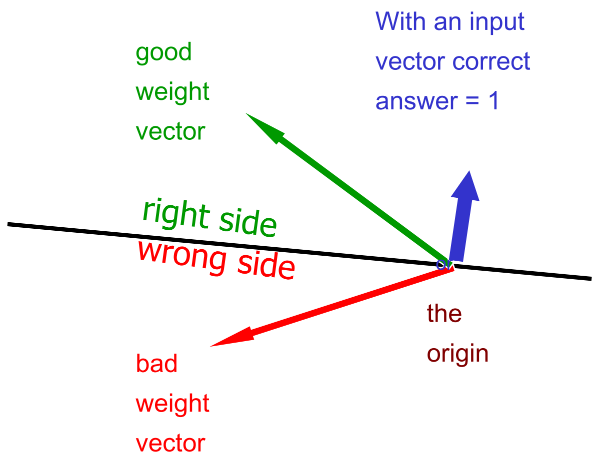


Each case defines a hyperplane:

Hyperplane perpendicular to the input vector. The weights should be one side of the hyperplane.

perceptrons

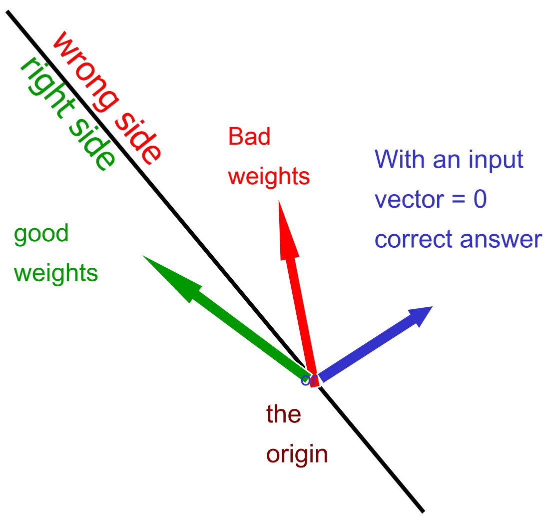
Learning algorithm: Geometric interpretation



If the dot product of the input vector with the weight vector is negative, the output will be wrong.

perceptrons

Learning algorithm: Geometric interpretation

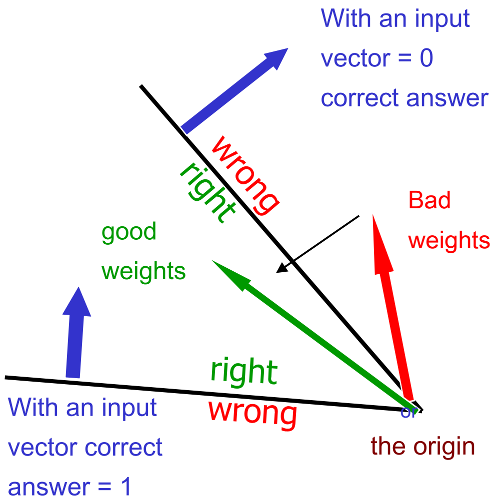


If the dot product of the input vector with the weight vector is negative, the output will be wrong.

perceptrons

Learning algorithm: Geometric interpretation

The hipercono of feasible solutions

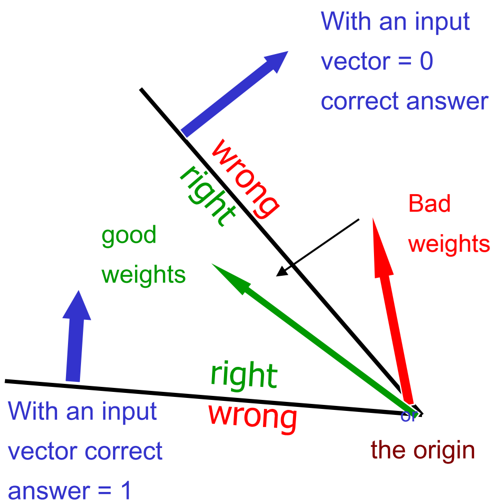


For learning is correct, we must find a point that is on the right side of all hyperplanes (which may not exist !!!).

perceptrons

Learning algorithm: Geometric interpretation

The hipercono of feasible solutions

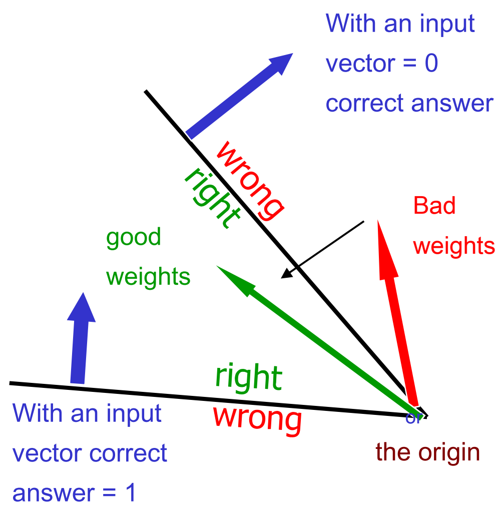


If there is a set of weights to provide the right answer for all cases, it will be in a hipercono with its apex at the origin.

perceptrons

Learning algorithm: Geometric interpretation

The hipercono of feasible solutions



In defining the feasible solutions hipercono a convex region, the average of two good weight vectors is also a good weight vector ...

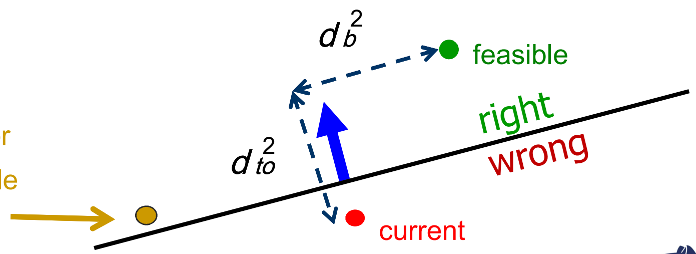
perceptrons

Learning algorithm: Correction

Algorithm

erroneous assumption: Each time the perceptron makes an error, the learning algorithm on the vector current pesos to all feasible solutions.

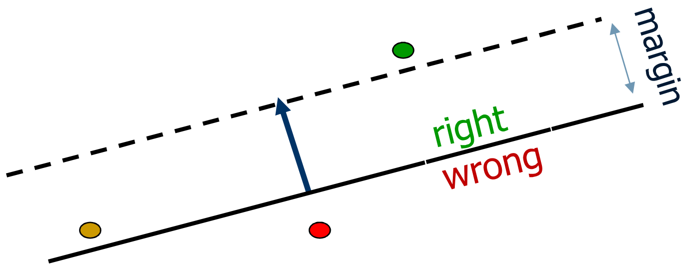
Problem case: The weight vector May not get closer to this feasible vector!



Learning algorithm: Correction

Algorithm

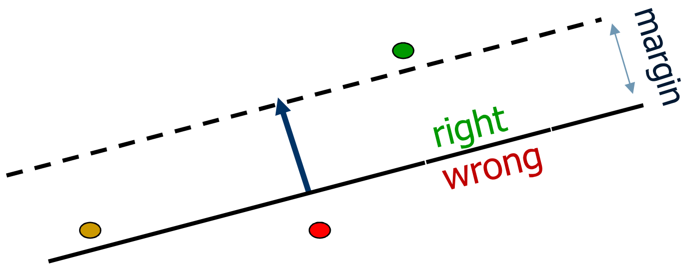
We consider vectors "generously feasible" weights remain within the feasible region, with a margin at least as large as the length of the input vector.



Learning algorithm: Correction

Algorithm

Each time the perceptron is wrong, the square of the distance all those vectors "generously feasible" weights always decrements in at least the square of the length vector updating the weights.

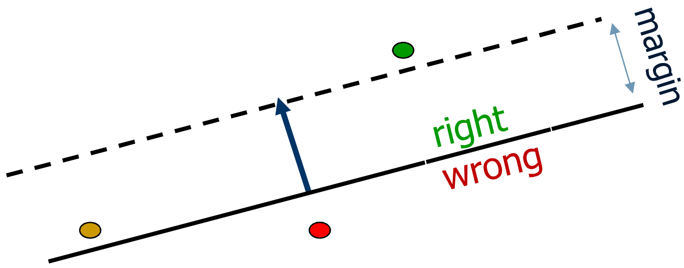


perceptrons

Learning algorithm: Correction

Algorithm

Therefore, after a finite number of errors, the weight vector should be in the feasible region, if present.



perceptrons

limitations

If you can choose all the features you want, you can do anything.

e.g. With an entry for each possible vector (2^n)
You can discriminate any Boolean function, although the Perceptron not generalize well.

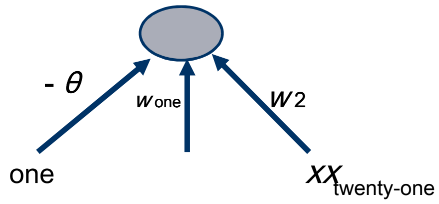
If the inputs are determined, there are severe limitations on what a perceptron can learn (eg XOR and EQ).

perceptrons

limitations

A perceptron can not decide if two bits are equal:

X one	X 2	Y
0	0	one
0	one	0
one	0	0
one	one	one



4 patients define four impossible odds to meet:

$w_1 + w_2 \geq \theta$, $0 \geq \theta$

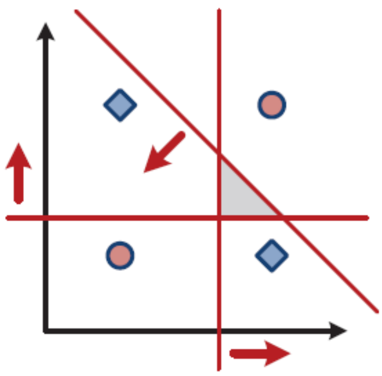
$w_1 < \theta$, $w_2 < \theta$

perceptrons

limitations

A perceptron can not decide if two bits are equal:

The XOR function defines
a system of inequations unsolvable:



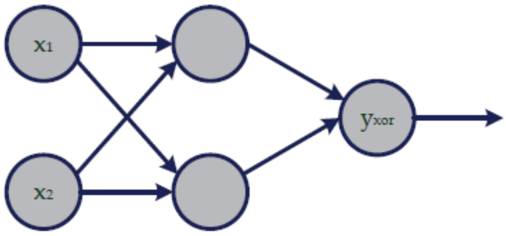
perceptrons

limitations

A perceptron can not decide if two bits are equal ...

... But we can do it with 2 layers of perceptrons:

Neurona	Entradas		Umbral θ
y_{00}	x_1	x_2	0.5
y_{11}	x_1	x_2	1.5
y_{xor}	$0.6y_{00}$	$-0.2y_{11}$	0.5



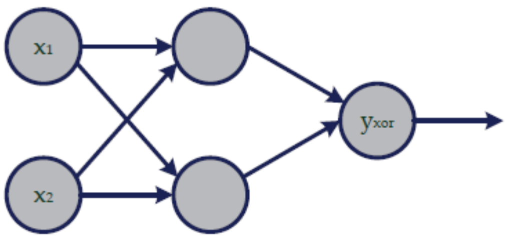
perceptrons

limitations

A perceptron can not decide if two bits are equal ...

... and also in several different ways:

Neurona	Entradas		Umbral θ
y_{10}	$2x_1$	$-1x_2$	2
y_{01}	$-1x_1$	$2x_2$	2
y_{xor}	$2y_{10}$	$2y_{01}$	2

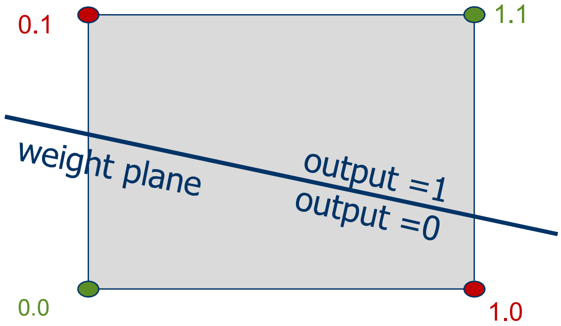


perceptrons

limitations:

geometric interpretation

positive and negative cases can not be separated by a plane



Data space:

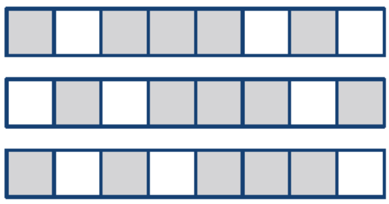
A dimension for each feature. An input vector is a point. A weight vector defines a hyperplane. The hyperplane is perpendicular to the vector of weights and is at a distance from the origin given by the threshold θ.

perceptrons

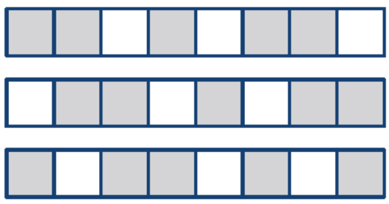
limitations

Differentiate between different patterns with the same number of pixels can not be allowed if translations:

TO



B



perceptrons

limitations

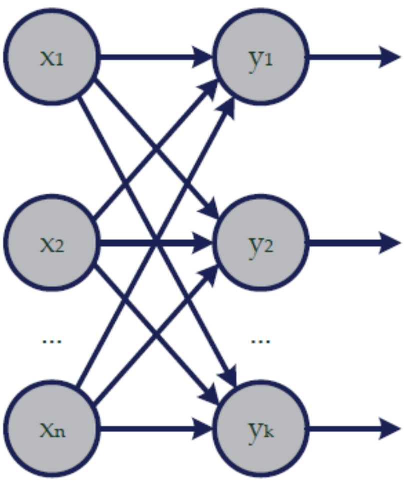
He **invariance theorem groups** Minsky and Papert provides that the portion perceptron learning is not able to recognize patterns if transformations that may be subject said patterns form a group.

The interesting part of pattern recognition must be resolved manually (by adding new features), but it can be learned using a Perceptron ...

perceptrons

The multiclass Perceptron

A perceptron for each kind of problem ...



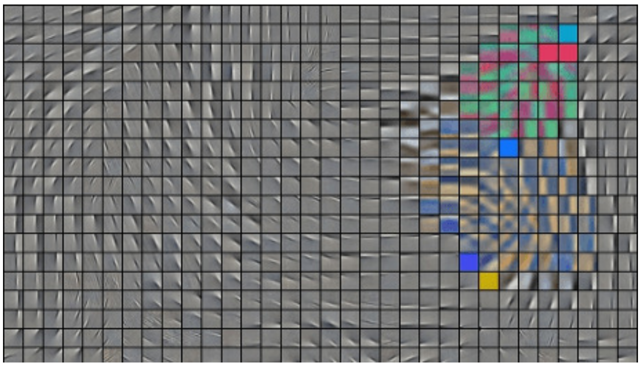
References

Neural Networks for Machine Learning

by Geoffrey Hinton

(University of Toronto & Google)

https://www.coursera.org/course/neuralnets



UNIVERSITY OF TORONTO

Bibliography

recommended reading

Fernando Berzal:

Deep Neural Networks & Learning

C HAPTER 7

perceptrons